12/09/2023

* Program for sorting all the negative numbers at left side →

I/P = -59, 47, 0, 39, -4, -63, 79, 90, -100

O/P = -59, -4, -63, -100, 47, 0, 79, 90, 39

Logic → Traverse a loop for i=0 & i<size
Initialise another variable J=0.
If arr[i]<0, then →
swap(arr[i], arr[J]) and J++;
Print array

Program →

```
#include<iostream>
using namespace std;
void organiseArray(int arr[], int size){
  int J=0;
 for(int i=0; i<size; i++){
    if(arr[i]<0){
      swap(arr[i], arr[J]);
      J++;
    }
 }
 for(int i=0; i<size; i++){
      cout<< arr[i]<<" ";
  }
}
int main(){
  int arr[]={-59,47,0,39,-4,-63,79,90,-100};
  int size = 9;
  organiseArray(arr, size);
}
```

* Program for sorting 0's, 1's and 2's →

I/P = 0, 1, 1, 2, 2, 0, 1, 2, 0

O/P = 0 0 0 1 1 1 2 2 2

logic → Initialize left = 0, index = 0 and right = size - 1.

Traverse a while loop for condition (index <= right)

Because, Jis position pr right indicate kar raha hoga, uske baad wale saare elements 2 honge. That's why we need not to check after that position.

Now, if (arr[index] == 0), then →
swap (arr[index], arr[left])
left++;
index++;

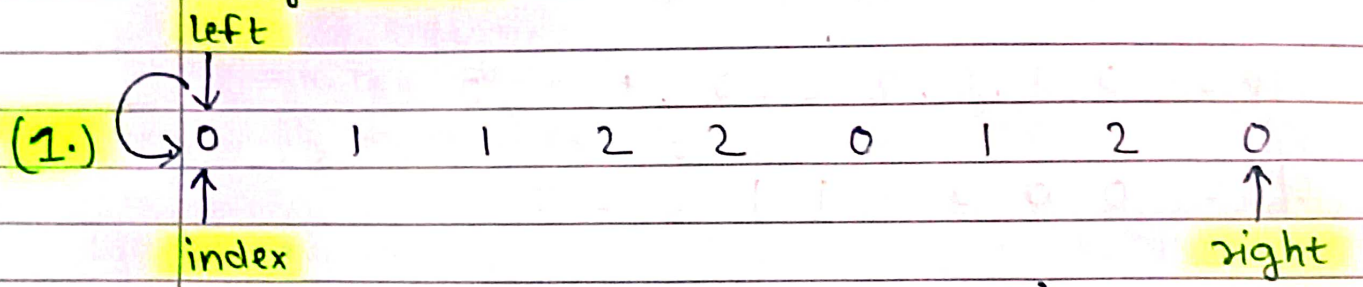
else if (arr[index] == 2), then →
swap (arr[index], arr[right])
right--;

// yha par index++ nahi karna hai because we don't know ki Jo element arr[right] position se swap hokar arr[index] position par aaya hai vo 0/1/2 hai. That's why, we have to check again for the element at that position and we will not increment index.

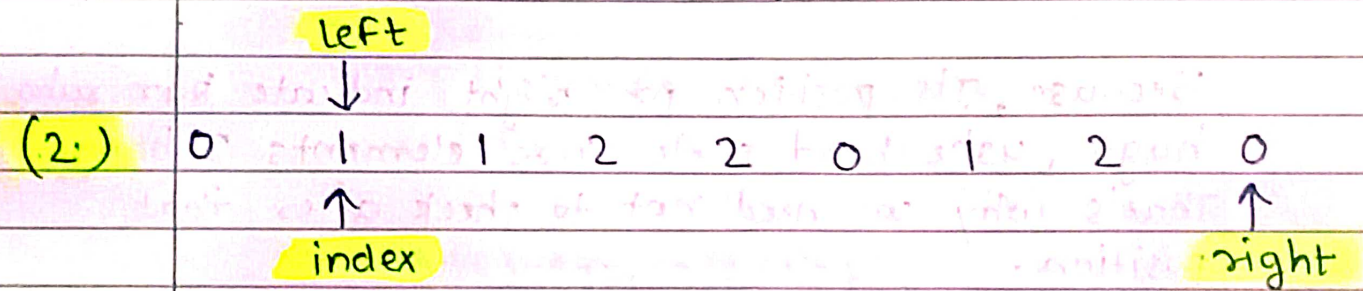
else, index++; (// Here, only element 1 is left)

__/__/__

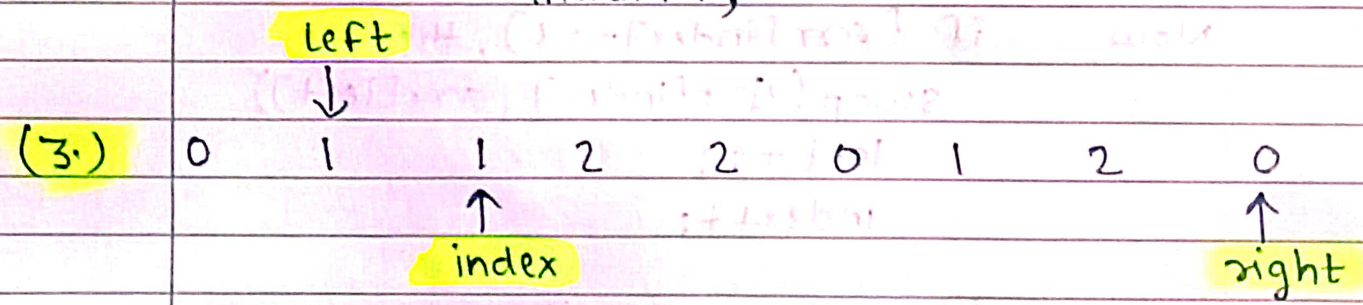
Dry Run →



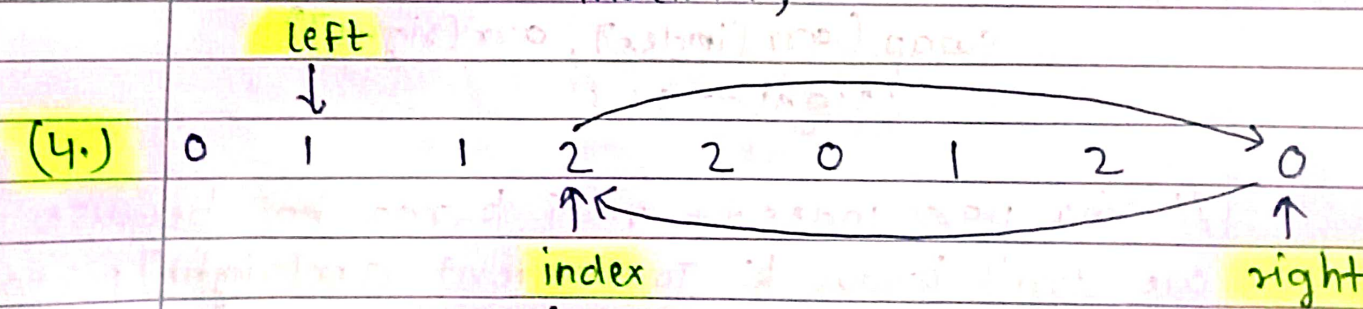
swap(arr[index], arr[left])
left++;
index++;



index++;

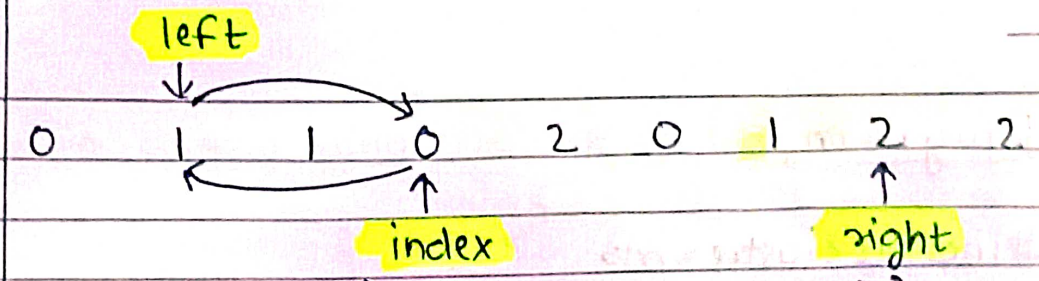


index++;



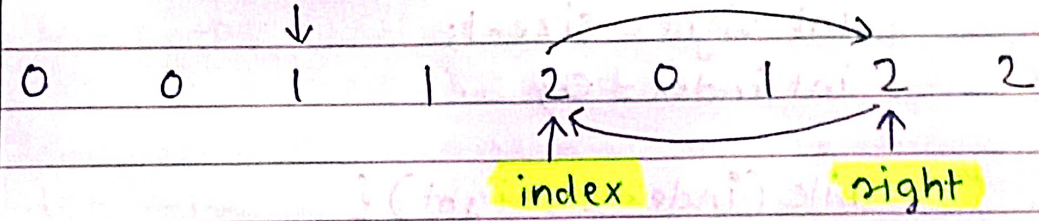
swap(arr[index], arr[right])
right--;

(5.)



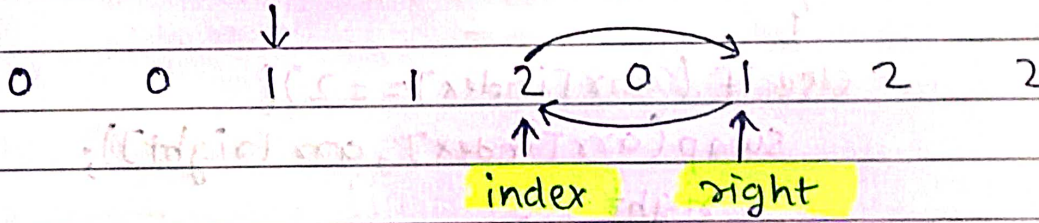
swap (arr[index], arr[left])
left++;
index++;

(6.)



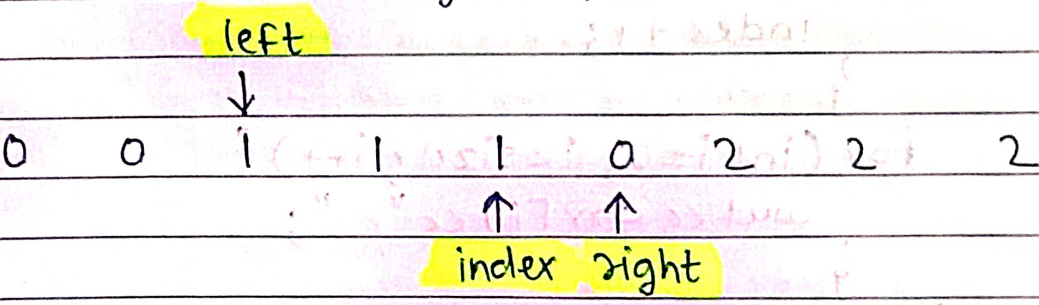
swap (arr[index], arr[right])
right--;

(7.)



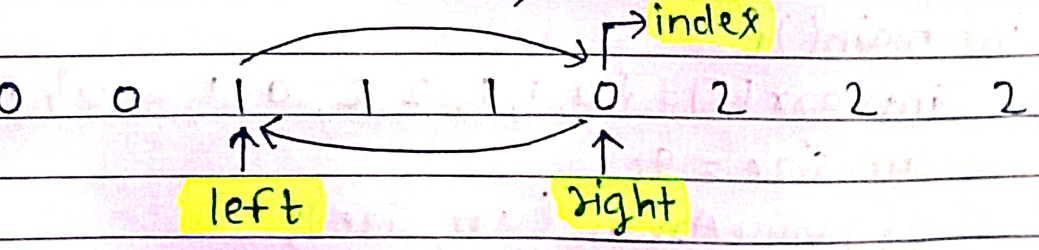
swap (arr[index], arr[right])
right--;

(8.)



index++;

(9.)



0	0	0	1	1	1	2	2	2

Sorted Array

__/__/____

Program →

```
#include<iostream>
using namespace std;

void organiseArray(int arr[], int size){
        int left = 0;
        int right = size-1;
        int index = 0;

    while (index<=right){
        if (arr[index]==0){
        swap(arr[index], arr[left]);
        left++; index++;
        }
    else if (arr[index]==2){
        swap(arr[index], arr[right]);
          right--;
        }
      else
        index++;
      }
   for (int i=0; i<size ; i++) {
        cout<< arr[i]<<" ";
      }
   }
int main(){
   int arr[]={0,1,1,2,2,0,1,2,0};
   int size = 9;
   organiseArray (arr, size);

   }
```

* **Program to shift array by k →**

I/P = 10, 20, 30, 40, 50, 60
k = 3

O/P = 40 50 60 10 20 30

Logic → Initialize a new array and take input k.
let k=3 and let new array be brr[6].
Traverse a loop for int index=0, index<n
Initialize a new variable newIndex = (index+k)%n
let index = 0, newIndex = (0+3)%6 = 3
index = 1, newIndex = (1+3)%6 = 4
and so on upto n-1.
brr[newIndex] = arr[index]
Print brr[]

Program →

```
#include <iostream>
using namespace std;
void shiftArray(int arr[], int n){
    int brr[6]; int k; cin>>k; int newIndex=0;
for (int index=0; index<n; index++){
        newIndex = (index+k)%n;
        brr[newIndex]= arr[index];
    }
for (int i=0; i<n; i++){
        cout<< brr[i]<<" ";
    } }
int main(){
    int arr[]={10,20,30,40,50,60};
    int n = 6;
    shiftArray(arr, n);
}
```

* Program to find missing number in array from range 0 → n.

I/P = 0 4 2 1 5 and n = 5

O/P = 3

logic → Initialize sum = 0. Traverse a loop for i=0, i<n.
sum = sum + arr[i].
Initialize another variable actualSum = 0.
actualSum = $\frac{n(n+1)}{2}$ } // Sum of n numbers in A.P.
Print actualSum - sum.

Program →

```
#include<iostream>
#include<vector>
using namespace std;
void missingNumber(vector<int>arr, int n){
    int sum=0;
    for(int i=0; i<n; i++){
        sum=sum+arr[i];
    }
    int actualSum=0;
    actualSum=(n*(n+1))/2;
    int ans= actualSum-sum;
    cout<<ans<<" ";
}
int main(){
    vector<int>arr{0,4,2,1,5};
    int n=5;
    missingNumber(arr,n);
}
```

* Given a mxn binary matrix mat, find the 0-indexed position of row that contains maximum count and number of ones in that row.

I/P =

row no.↓	0	1	2	3 → column no.
0	0	1	0	1
1	0	1	1	1
2	1	0	0	0

O/P = max one count is in row: 1
max one count is: 3

logic → Initialize a variable oneCount = INT_MIN
Initialize a variable rowNo. which keeps the track of row with maximum 1's.
Traverse an outer loop for row.
Initialise a variable count = 0 before the inner loop for columns. The count variable will make the count zero after traversing each row.
Check condition if mat[i][j] = 1
increment the count

After traversing each row fully,
check condition if count > onecount
if yes, assign oneCount = count;
rowNo. = i

Print the rowNo. and oneCount.

Program→

```
#include<iostream>
#include<vector>
#include<limits.h>
using namespace std;

void maxOneCount(vector<vector<int>>mat){
    int i=0;
    int m= mat.size();
    int n= mat[i].size();
  // it will count no. of 1's in each row
    int oneCount=INT_MIN;
  // it will tell which row no. has max 1's count
    int rowNo=0;
  for(int i=0; i<m ; i++){
      int count=0;
    for(int j=0 ; j<n; j++){
      if(mat[i][j]==1){
          count++;
      }
    }
   if(count>oneCount){
       oneCount=count;
       rowNo=i;
    }}
cout<<"max one count is in row:"<<rowNo<<endl;
cout<<"max one count is:"<<oneCount<<endl;
}
int main(){
  vector<vector<int>>mat{
   {0,1,0,1},{0,1,1,1},{1,0,0,0}};

  maxOneCount(mat);
}
```

__/__/__

* Given an nxn 2D matrix, representing an image, rotate the image by 90° (clockwise) →

I/P = 1 2 3
4 5 6
7 8 9

O/P = 7 4 1
8 5 2
9 6 3

Logic → Transpose the input matrix.

Traverse the outer loop for row i=0, i<n
n = mat.size( ) i.e. no. of rows in matrix
Traverse the inner loop for column J=i, J<mat[i].size()

```
swap(mat[i][J], mat[J][i])
```

Reverse the transposed matrix.

We have to reverse the rows of the matrix.
So, there will be single loop only for i=0, i<n

```
reverse(mat[i].begin(), mat[i].end());
```

Print the resultant matrix.

Program→

```
#include <iostream>
#include <vector>
#include <algorithm> // header file to include reverse function

using namespace std;

void rotate90(vector<vector<int>> mat){
    int n= mat.size();
    //transpose
    for(int i=0; i<n; i++){
        for(int j=i; j<mat[i].size(); j++){
            swap(mat[i][j],mat[j][i]);
        }
    }
    //reverse
    for(int i=0; i<n; i++){
        reverse(mat[i].begin(),mat[i].end());
    }
    // printing
    for(int i=0; i<n; i++){
        for(int j=0; j<mat[i].size(); j++){
            cout<<mat[i][j]<<" ";
        }
        cout<<endl;
    }
}
int main(){
    vector<vector<int>>mat{{1,2,3},{4,5,6},{7,8,9}};

    rotate90(mat);

}
```